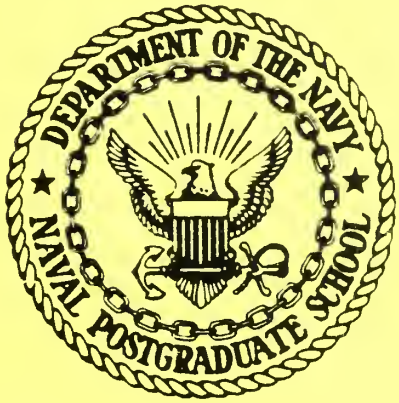
NPS-54-82-005

NAVAL POSTGRADUATE SCHOOL

Monterey, California



USABILITY OF MILITARY STANDARDS FOR THE MAINTENANCE OF EMBEDDED COMPUTER SOFTWARE

Norman F. Schneidewind

June 1982

Final Report: 1 Jan 82 to 1 Jun 82

Approved for public release; distribution unlimited.

Prepared for:
The Trident Command and Control Systems
Maintenance Agency
Newport, Rhode Island

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

David R. Schrady
Acting Provost

The work reported herein was supported by the Trident Command and Control Systems Maintenance Agency.

This report was prepared by:

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Date Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NPS-54-82-005	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: USABILITY OF MILITARY STANDARDS FOR THE MAINTENANCE OF EMBEDDED COMPUTER SOFTWARE	5. TYPE OF REPORT & PERIOD COVERED: Final Report, 1 Jan 82 to 1 Jun 82
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Norman F. Schneidewind	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: N0036 78PON3907
11. CONTROLLING OFFICE NAME AND ADDRESS: Trident Command and Control Systems Maintenance Agency, Newport, Rhode Island	12. REPORT DATE
	13. NUMBER OF PAGES: 22
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

A7-E Aircraft software redesign project	Software design
Microcomputer software	Software maintainability
Military Standard MIL-STD 1679	Software maintenance
Requirements analysis methodologies	Software standards
SECNAVINST 3560.1	Traceability

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Several military software standards were examined and evaluated with respect to their applicability and usability for maintaining embedded computer software. These standards included the following: Department of the Navy Tactical Digital System Documentation Standards, SECNAVINST 3560.1; MIL-STD 1679, Navy Military Standard for Weapon System Development; and Weapon Specification 8506. These standards were discussed from three standpoints: (1) the degree to which they support the use of newer software development

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

19. Weapons Specification WS 8506

20. technologies (e.g., requirements analysis methodologies) for improving software maintenance; (2) the effect of the microcomputer and its software development environment on the application of these standards; and (3) the extent to which these standards enhance traceability (tracing the various levels of related documentation). These aspects required a reevaluation of the applicability of software standards. A recommendation is made to use the A7-E Aircraft software redesign project as a model for improving (1) and (3) in the three standards. Item (2) was judged to be not relevant to the development of software standards.

TABLE OF CONTENTS

1. INTRODUCTION

This paper addresses the question of how useful military software standards are for maintaining embedded computer software. Our discussion builds on previous studies of this topic (Schneidewind, 1982) which involved an analysis of the following United States Navy publications:

- Military Standard (MIL-STD) 1679;
- Weapons Specification (WS) 8506; and
- Tactical Digital Systems Documentation Standards, SECNAVINST 3560.1.

(Note: It is recognized that, technically, only the first document is a standard. For ease of exposition, all three are referred to as "standards" in this paper.) The question posed by the previous research study was: Could these standards, accompanied by basic program documentation, such as a listing, provide adequate guidance for a new programmer to maintain software, such as that found in the Trident Command and Control Subsystem? These standards were reviewed with respect to the following criteria:

- design approaches for achieving good maintainability;
- specification and documentation requirements for achieving good maintainability; and
- testing approaches for achieving good maintainability.

With some significant qualifications, it was concluded that these standards were adequate for maintenance purposes. However, it was pointed out that these standards were developed for use in design and not for maintenance specifically. (The interested reader may find the details in the references

which have been cited.)

Now, an excellent standard would recognize the linkage between software design and maintenance and would specify design practices that are conducive to maintenance. The problem seems to be that standards of the type which have been referenced were developed prior to the time when maintenance was recognized as an important phase of the software life cycle and prior to the realization that maintainability must be designed into the software. Software standards should be revised to reflect this important concept. Also, advances in software requirements analysis and design methodologies, coupled with the significant use of microcomputers in embedded computer systems, have led to the need to update military software standards to reflect the realities of newer design and programming environments. Improvement in design approach enhances maintainability; the use of microcomputers, on the other hand, presents new problems for the software development agency due to the limited software development tools which are available in many microcomputer software development facilities. However, it is an open question as to whether the increased use of microcomputers for embedded systems is aiding or retarding the production of maintainable software. Although many microcomputer software production facilities are low-level, oriented to assembly language programming, the trend is for microcomputer software to be developed on larger host machines, using elaborate program development tools on an interactive basis, and down loaded to a development system and eventually to the target machine (Ziegler, 1982). This trend may lead in the future to more emphasis on chip certification and less on software validation. As contrasted to

advances in software design and programming methodology, it is not clear that software standards should be significantly changed just because computers get smaller and programming environments change. The desirable objectives of, for example, producing code which can be changed without upsetting the rest of the system remains valid independent of the particular form, size, speed or configuration of a hardware-software system.

Transcending the aspects of improved software design methodology and changing computer technology, is the need to trace both errors in the software and design decisions (which many times are related to errors) to the pertinent technical information. The need for traceability is independent of software design methodology and the particular computer technology which is used in a system. However, proper use of methodology and technology can greatly improve traceability, particularly with regard to identifying the effects of changes to the software.

Within the context of the question posed at the beginning of this section, we examine the usability of military software standards in the ensuing sections with respect to the following areas:

- requirements analysis methodologies;
- microcomputer software development; and
- traceability.

We conclude with recommendations concerning the effective utilization of these standards in an environment of changing methodology and technology.

2. EFFECTS OF REQUIREMENTS ANALYSIS SYSTEMS ON SOFTWARE STANDARDS

One of the major efforts to improve the quality of software has focused on the development of formal software requirements analysis methodologies (Alford, 1977; Bell, 1977; Ross, 1977; Teichroew, 1977). Objectives of these and related systems are the following:

- improved quality of documentation with regard to precision, consistency and completeness;
- formal methods of specifying requirements, usually involving the use of a language or format for expressing requirements (Liskov, 1975); and
- separation of system functions so that related functions appear in the same module and unrelated functions appear in different modules, resulting in the creation of independent modules (Myers, 1978).

A major ingredient of requirements analysis methodologies is computer-aided analysis, consisting of the following components: a language for expressing requirements; a data base for storing requirements and specifications; an analysis and retrieval system for checking requirements consistency and completeness; and various types of graphics terminal and hardcopy outputs (Bell, 1977). The emphasis of these systems is a language aimed at achieving formalism and consistency of expressing requirements. These methodologies do not address to a great extent strategies for translating requirements into a software design.

An effort where design strategies and requirements analysis techniques are directed toward confining the effects of changes to software (hence,

improving maintainability) is the project of the Naval Research Laboratory (Britton, 1981; Heninger, 1978, 1980; Parker, 1980) to rewrite the software for the A-7E Aircraft, using the principles of information hiding, separation of concerns and abstract interfaces (Parnas, 1972, 1978; Hestor, 1981; Britton, 1981). Central to this effort is the method for decomposing modules. The method proposed by Parnas (Parnas, 1972) is the following:

- every module in the decomposition process is characterized by design decisions which are hidden from all other modules (the information hiding principle); this criterion does not decompose the system into modules on the basis of the time sequence of processing the modules; and
- elements that are likely to change are identified and incorporated into separate modules (device interface modules) in order to minimize the effects of device changes on user modules.

Myers (Myers, 1978), among others, has sounded a similar theme. He recommends that modules should be partitioned so that relationships between modules are minimized. In Myer's terms, this results in high module strength and weak module coupling, and leads to the desirable result of module independence. A major objective of these approaches is to reduce the ripple effect of software changes (that is, the effect on modules external to the changed module).

2.1 STATUS OF STANDARDS RELATIVE TO REQUIREMENTS ANALYSIS SYSTEMS

What is the status of the standards (1679, 8506, 3560.1) relative to specifying the use of requirements analysis methodologies and design techniques (e.g., methods for module decomposition)? MIL-STD-1679, Section 5.2, states that the design shall be a hierarchical structure with the highest level of control logic residing at the top of the hierarchy and computation functions residing at the lower levels. As stated by Myers (Myers, 1978), the objective is not simply partitioning modules into a hierarchy, but partitioning so that each module is as independent of other modules as possible. This procedure will result in confining the effects of change and, hence, make the software more maintainable. In addition, as indicated by Schneidewind (Schneidewind, 1982, NPS-54-82-002), although the change in reporting and control procedures specified by 1679 is excellent in Section 5.11.2, the coverage is inadequate regarding separation of software functions by anticipated degree of change. With regard to WS 8506, it makes brief mention of describing major functions and the dependency among functions in Section 5.2. This reference does not elaborate on why or how the information is to be used (Schneidewind, 1982, NPS-54-82-002). An important use of the information would be for decomposing a system into modules and for the related purpose of achieving module independence. The situation is worse in the case of SECNAVINST 3560.1. This document is notable for the great amount of detail presented pertaining to function and interface descriptions, data exchange, program resource budgets, etc. (Schneidewind, 1982, NPS-59-82-003). However,

there is an absence of material dealing with designing for change.

In addition to the above gaps in coverage, the standards pre-date the use of software requirements analysis systems, such as those described by Alford (Alford, 1977). Naturally, the older the standard, the more obsolete it is relative to advances in requirements analysis methodologies and software design. So these remarks should not be construed as criticism of the standards, but as indications of the need to consider updating the standards for the purpose of bringing them into line with software engineering techniques which look quite promising. It is necessary to emphasize that methods proposed by Parnas, Myers, Teichroew and others have not reached the stage of accepted practice by a large segment of the software engineering community. For one thing, there must be further demonstration of improvements in software maintainability on large systems before these procedures will graduate to the status of standard practice. However, we contend that the approach in standards development should be to lead rather than to follow developments in software engineering. There is obviously more risk associated with this policy, but its successful implementation can prevent a standard from being obsolete before it is even issued.

In summary, with regard to the areas of requirements analysis and software design, the standards are weak and in need of upgrading to include the following requirements:

- decomposition of a system into modules for the purpose of confining the effects of software changes by using techniques such as:
  - hiding device characteristics from user programs and,
  - designing modules so that related elements are contained in

the same module and unrelated elements are contained in different modules (high module strength and low module coupling); and

° employment of a requirements analysis system for the purposes of:

- standardizing the language in which requirements are stated, and
- providing computer-aided tools for storing, analyzing and retrieving requirements to ensure consistency and completeness.

3. THE EFFECT OF MICROCOMPUTERS ON STANDARDS DEVELOPMENT

As suggested in Section 1, standards development should be independent of the characteristics of the hardware and software employed in an application. A standard should require the developer to employ sound software engineering practices. These techniques become 'sound' by evolving from theory to standard practice through a process of proposal, debate, demonstration, use, concensus and acceptance. This process should not be influenced by whether, for example, an application is implemented in a centralized main frame or a distributed microcomputer system. The aspect that is affected by the choice of technology is the ability of the developer to meet the standard (e.g., conforming to a requirement for using structured programming with assembly language versus high level language). For example, if crude program development and test tools are used for implementing microcomputer software, or any software for that matter, traceability will be difficult to achieve. More will be said about traceability in a later section.

Concern about the peculiarities of microcomputer software development may evaporate in the near future. As mentioned in Section 1, the trend in microcomputer software development is toward using the types of tools which have been used for some time in the minicomputer and mainframe areas. For example, if we compare two publications which are only one year apart (Ogdin, 1980) and (Markowitz, 1981), we find that the former, in describing microcomputer programming environments, stresses hexidecimal coding, prototyping systems, computer evaluation kits, portable front panels, single board

computers and microcomputer development systems. In contrast, Markowitz's article describes the architecture of the iAPX 286 in terms of memory management, segmented memory, protection, and various priviledge levels - characteristics of large machines. Zeigler (Zeigler, 1981) talks about extensions to the ADA language which INTEL has developed in support of the 432 architecture.

None of the standards makes reference to the use of microcomputers. This would obviously be the case for 3560.1 and 8506, since their publication pre-dated significant use of microcomputers. Although 1679 was published during the era of microcomputers, mentioning the use of this technology in the standard would have been inappropriate. As stated by Cooper (Cooper, 1981), in describing the development of 1679, the single most important rule of a MIL-STD is that it can only specify what is required, not how to satisfy a requirement. However, it must be noted that 1679 does not seem to be entirely faithful to this rule, since it calls for the use of structured programming constructs (Section 5.3), and top down design and high order languages (Section 5.5), as examples of many "how to" provisions.

Based on the reasons given in this section, we conclude that the standards should not be modified to incorporate provisions that deal with the development of microcomputer software for embedded computer systems.

4. APPROACHES FOR IMPROVING TRACEABILITY OF STANDARDS

A prerequisite for achieving traceability and, hence, maintainability, is to have planned for the software to change when it was designed and to have designed the software correspondingly, so that changes can be easily traced through the documentation in order to identify the relevant inputs, outputs, data base, modes of operation, conditions, etc. The A-7E Aircraft documentation (Heninger, 1978) does a good job of providing traceability because it was designed with change in mind. Some of the formats which are useful for achieving traceability are the following:

- Event Tables which relate modes, events, and actions;
- Condition Tables which relate modes, conditions and actions or values; and
- Selector Tables which relate modes and mutually exclusive characteristics of modes.

In the above, the following meanings apply:

- mode - system state;
- condition - expression whose value is true or false and characterizes the system for a measurable time;
- event - a condition which changes from true to false or vice versa at a specific moment in time;
- action - evaluation of a function; and
- value - expression or output data item value.

In this system of documentation, if an event (e.g., ground distance to a reference point) were to change when the radar update mode is entered,

it would be possible to ascertain the fact that this combination affects an action taken by the pilot relative to cursor enable (output data item). In general, design decisions, module decomposition and module dependencies are made explicit in the system of software design and documentation. Other useful aspects of the documentation include a dictionary of commonly used terms and a section dealing with subsets - a part of the system which is isolatable from the total system, performs part of the services provided by the total system and uses less computer resources than the total system. One of the ideas of subsets is to be able to reassemble a smaller system and thereby save on resources if the entire system is not utilized (e.g., a particular weapon is not available or used).

Weaknesses in this system seem to lie in the areas of tracking changes in outputs to inputs and data bases, where applicable, and, in some cases, lack of clarity of definitions as they relate to various tables. Also, although we subscribe to the objectives of information hiding, which provides the underpinning of the A-7E project, it is our opinion that the design procedures and terminology which are necessary to implement information hiding could be confusing to software engineers. We prefer to think of the process of requirements analysis as making requirements explicit, rather than hiding certain ones, and to only embody a requirement in a module when the requirement is relevant to that module; otherwise, the requirement is implemented in a module where it is relevant. Requirements which are common to two or more modules are contained within a separate module, rather than within the given modules themselves. Additionally, requirements which are likely to change should be quarantined

and placed in a limited number of modules rather than being spread across many modules.

Nevertheless, on the whole, the A-7E Aircraft project and a related project (Hester, 1981) would provide an excellent model for revising the standards to incorporate software design practices which specifically address the need to account for future change to the software. This would be a particularly powerful approach if coupled with the use of one of the requirements analysis systems (Ross, 1977) for providing computer-aided requirements analysis tools and for supporting the requirements analysis which must precede the software design.

The primary author of MIL-STD 1679 (Cooper, 1981) feels that with only two years use, it would be premature to revise it. However, neither 1679 nor the other standards are strong in the vital area of traceability (Schneidewind, 1982). We feel, therefore, that because the volatility of software is so great and affects maintenance so significantly, that a standard must explicitly provide for change in the design process in order to achieve traceability in the maintenance phase. Note that this characteristic of a standard is not the same thing as a change control procedure, which is a part of 1679. To use a medical analogy, the recommended approach involves using preventive medicine early in the life of a system in order to avoid emergency surgery at a later date.

5. CONCLUSIONS AND RECOMMENDATION

Three important areas relative to software standards have been considered which potentially impact on software maintainability:

(1) requirements analysis and software design methodologies;

(2) microcomputer software; and

(3) traceability.

It is concluded that (1) and (3) should be improved in SECNAVINST 3560.1, WS8506 and MIL-STD 1679 and that (2) is not appropriate for inclusion in a standard.

Furthermore, it is recommended that A7-E Aircraft software redesign project be used as a model for improving the standards relative to (1) and (3).

REFERENCES

Alford, M.W., Jan. 1977, "A Requirements Engineering Methodology for Real-Time Processing Requirements", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, pp. 60-69.

Bell, T.E., Bixley, D.C. and Dyer, M.E., Jan. 1977, "An Extendable Approach to Computer-Aided Software, Requirements Engineering", Jan. 1977, IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, pp. 49-60.

Britton, K.H. and Parnas, D.L., Dec. 1981, "A-7E Software Module Guide", NRL Memorandum Report 4702, Naval Research Laboratory, Washington, D.C.

Britton, Kathryn Heninger, Parker Alan R. and Parnas, David L., Mar. 1981, "A Procedure for Designing Abstract Interfaces for Device Interface, Modules", Proceedings of the 5th International Conference on Software Engineering, San Diego, CA pp. 195-204.

Cooper, Jack, Aug. 1981, "Development of MIL-STD-1679, Software Engineering Standards Application Workshop", San Francisco, CA pp. 139-143.

Heninger, Kathryn, L., Kallander, John W. and Shore, John E., Nov. 1978, "Software Requirements for the A-7E Aircraft", NRL Memorandum Report 3876, Naval Research Laboratory, Washington, D.C.

Heninger, K.L., Jan. 1980, "Specifying Software Requirements for Complex Systems: New Techniques and Their Applications", IEEE Transactions on Software Engineering, Vol. SE-6, No. 1, pp. 2-13.

Hester, S.D., Parnas, D.L. and Ulter, D.F., Oct. 1981, "Using Documentation as a Software Design Medium", The Bell Systems Technical Journal, Vol. 60, No. 8, pp. 1941-1977.

Kahn, Kevin C. and Pollack, Fred, Feb. 1981, "An Extensible Operating System for the Intel 432", Digest of Papers, Spring COMPCON 81, San Francisco, CA., pp. 398-404.

Liskov, Barbara, H. and Zelles, Stephen N., Mar. 1975, "Specification Techniques for Data Abstractions", IEEE Transactions on Software Engineering, Vol. SE-1, No. 1, pp. 7-19.

Markowitz, R., Feb. 1981, "Software Impact on Microcomputer Architecture; A Case Study", Digest of Papers, Spring, COMPCON 81, San Francisco,CA., pp. 40-48.

MIL-STD-1679 (NAVY), Dec. 1978, "Military Standard, Weapon System Software Development", Department of Defense, Washington, D.C.

Myers, Glenford Jr., 1978, "Composite Structured Design", Van Nostrand Reinhold Company, New York, N.Y.

Ogdin, Carol, Anne, 1980, "Microcomputer Management and Programming", Prentice-Hall, Englewood Cliffs, N.J.

Parker, Robert A. et al., Nov. 1980, "Abstract Interface Specifications for the A-7E Device Interface Module", NRL Memorandum Report 4385, Naval Research Laboratory, Washington, D.C.

Parnas, D.L., Dec. 1972, "On the Criteria To Be Used in Decomposing Systems into Modules", Communications of the ACM, Vol. 15, No. 12, pp. 1053-1058.

Parnas, David L., May 1978, "Designing Software for Ease of Extension and Contraction", Proceedings of the 3rd International Conference on Software Engineering, Atlanta, GA, pp. 264-270.

Ross, D.T. and Schoman, K.E., Jr., Jan. 1977, "Structured Analysis for Requirements Definition", IEEE Transactions of Software Engineering, Vol. SE-3, No. 1, pp. 6-15.

Schneidewind, N.F., Feb. 1982, "Software Maintenance: Improvement Through Better Development Standards and Documentation", Naval Postgraduate School, NPS-54-82-002, Monterey, CA.

Schneidewind, N.F., Feb. 1982, "Evaluation of SECNAVINST 3560.1 Tactical Digital Systems Documentation Standard For Software Maintenance", Naval Postgraduate School, NPS-54-82-003, Monterey, CA.

Schneidewind, N.F., Feb. 1982, "Evaluation of Maintainability Enhancement for TCP/TSP Revision 6.0 Update .20", Naval Postgraduate School, NPS-54-82-004, Monterey, CA.

SECNAVINST 3560.1, 8 Aug. 1974, "Tactical Digital Systems Documentation Standard", Department of the Navy, Office of the Secretary, Washington, D.C.

Teichroew D. and Hershey, E.A., III, Jan. 1977, "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing". IEEE Transactions on Software Engineering, Vol. 1, pp. 41-48.

Weapons Specification, WS-8506, Rev 1, Nov. 1971, Requirements for Digital Computer Program Documentation, Naval Ordinance Systems Command, Department of the Navy, Washington, D.C.

Zeigler, Stephen, et. al., Feb. 1981, "The Intel 432 Ada Programming Environment", Digest of Papers, Spring COMPCON 81, San Francisco, CA., pp. 405-410.

DISCLAIMER

The opinions expressed in this report are strictly those of the author and do not necessarily reflect the opinions of the Naval Postgraduate School, Department of the Navy, or Department of Defense.

DISTRIBUTION LIST

	No. Copies
Prof. Victor Basili Department of Computer Science University of Maryland College Park, MD 20742	1
Mr. Laszlo Belady IBM Corporation (VAL) Old Orchard Road Armonk, NY 10504	1
Dr. Barry Boehm Software Research and Technology Defense and Space Systems Group TRW One Space Park Redondo Beach, CA 90278	1
Dr. Ned Chapin Info Sci. Inc. Box 7117 Menlo Park, CA 94025	1
Mr. John Cooper Anchor Software 4111 Century Court Alexandria, VA 22312	1
Prof. Lyle Cox Code 52C1 Computer Science Department Naval Postgraduate School Monterey, CA 93940	1
Mr. Barry Deroze TRW/DSSG Mail Station R2 1410 One Space Park Redondo Beach, CA 90278	1
Mr. William Ferreira Trident CCSMA Building 12 U.S. Navy Newport, RI 02840	1

Dr. Matthew Fisher
No. 1 Violante Court
Eatontown, NJ 07724 1

Mr. Jan Fränlund
TELUB AB
Ljungadalsgatan 2, Box 1232
SE-351 12
Växjö, Sweden 1

Mr. Tom Gilb
Infotect
Iver Holters Vei 2
N-1410
Kolbotn, Norway 1

Mr. Gerald Goulet
Naval Air Development Center
Warminister, PA 18974 1

Dr. Robert Grafton
Code 437
Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217 1

Ms. Kathryn Heninger
Naval Research Laboratory
Information Processing Systems Branch
Communications Science Division
Washington, DC 20375 1

Prof. Melvin Kline
Code 54Kx
Administrative Sciences Department
Naval Postgraduate School
Monterey, CA 93940 1

Mr. Robert G. Lanergan
Raytheon Company
Missile Systems Division
Hartwell Road
Mail Stop BLA 1-4
Bedford, MA 01730 1

Prof. Bennet Lientz
Graduate School of Management
University of California at Los Angeles
Los Angeles, CA 90032 1

LTC Casper H. Lucas 1
HQ, AFLCILOEC
Wright-Patterson AFB, OH 45431

Prof. Norman Lyons 1
Code 54Lb
Administrative Sciences Department
Naval Postgraduate School
Monterey, CA 93940

Mr. Paul A. Mauro 1
Hughes Aircraft Company
Bldg. 618, Mail Stop B218
P.O. Box 3310
Fullerton, CA 92634

Mr. Jim McCall 1
GE Space Division
1277 Orleans Drive
Sunnyvale, CA 94086

Dr. Edward Miller 1
Software Research Associates
P.O. Box 2432
San Francisco, CA 94126

LCDR Ronald W. Modes 1
Code 54Mf
Computer Science Department
Naval Postgraduate School
Monterey, CA 93940

Mr. John B. Munson 1
Corporate Software Engineering
System Development Corp.
2500 Colorado Ave.
Santa Monica, CA 90406

Mr. John Musa 1
Bell Telephone Laboratories
Rm. 3A332
Whippany Rd.
Whippany, NJ 07981

Dr. Peter Neumann 1
SRI International, EL 301
Menlo Park, CA 94025

Mr. Steve Oxman 1
Trident CCSMA
Building 132T
U.S. Navy
Newport RI 02840

Mr. Richard Pariseau 1
Naval Air Development Center
Warminister, PA 18974

Dr. David Parnas 1
Information Processing Systems Branch
Communications Science Division
Naval Research Laboratory
Washington, DC 20375

Prof. C.F. Ramamoorthy 1
Department of Electrical Engineering
and Computer Science
University of California
Berkeley, CA 94720

Prof. Norman F. Schneidewind 10
Code 54Ss
Administrative Sciences Department
Naval Postgraduate School
Monterey, CA 93940

Dr. John Stancil 1
Trident CCSMA
Building 132T
U.S. Navy
Newport, RI 02840

Dr. Leon Stucki 1
Boeing Computer Services
Seattle, WN 98124

Mr. David M. Weiss 1
Information Processing System Branch
Communications Science Division
Naval Research Laboratory
Washington, DC 20375

Prof. Roger Weissinger-Baylon 1
Code 54Wr
Administrative Sciences Department
Naval Postgraduate School
Monterey, CA 93940

Prof. Steven Yau 1
Department of Computer Science
Northwestern University
2001 Sheridan Rd.
Evanston, IL 60201

Prof. Marvin Zelkowitz 1
Department of Computer Science
University of Maryland
College Park, MD 20742

AS/OR Library 1
Code 54/55
Naval Postgraduate School
Monterey, CA 93940

Computer Center Library 1
Code 0141
Naval Postgraduate School
Monterey, CA 93940

Computer Science Department 1
Code 52
Naval Postgraduate School
Monterey, CA 93940

Defense Technical Information Center 1
Cameron Station
Alexandria, VA 23314

Knox Library 1
Code 0142
Naval Postgraduate School
Monterey, CA 93940

Office of Research Administration 1
Code 012A
Naval Postgraduate School
Monterey, CA 93940